PROJECT DOCUMENTATION

CS 165 : Database Systems
Simple Database Application

Developed by:
Don Rodolfo P. Abril
Student # 2014-63818
CS 165 WFR/FVW

Instructor:
Ms. Marie Jo-anne Mendoza

Application available through:
https://latw.herokuapp.com

GitHub repository:
https://github.com/dpabril/cs165dbapp

Table of Contents

I. Project Description

A. Background

This project, simply titled **Languages Around the World**, is a database of human languages spoken and either written or unwritten around the world. I conceptualized the project after discovering a considerable interest in foreign languages and geography, all while taking up a German language course in the university. Languages, the families they belong to, the scripts in which they are written in, and the countries whose peoples speak them are the details that are present in the application.

B. Objectives

Languages Around the World was made to:

1) Be an application giving basic information on languages, their dialects, language families, scripts, and countries
2) Assist in mapping the prevalence and use of languages in various locations around the world

And on more personal notes, this application was made:

3) To put into practice concepts learned in database systems and software engineering
4) To further explore my newfound interests in languages, linguistics, orthography, and geography

C. Scope

There are thousands of languages that are in use around the world. In addition, there are tens of language families, hundreds of scripts or writing systems, and over a hundred countries. To map this overwhelming amount of information is not practical for the purposes of the application as an undergraduate project on databases, and as such, only a very small subset of this information will be realized. Having said this, it would be an overestimation to take Languages Around the World as an alternative for existing sites that most definitely offer more comprehensive information on languages.

D. Sources

Languages Around the World sourced/sources information from Ethnologue, Glottolog, and Wikipedia.

II. Technical Specifications

A. Database Management System

Languages Around the World uses **PostgreSQL** for both development and production. The latter DBMS was chosen since it can be provided free of charge by Heroku which hosts my application.

B. Development Frameworks and Dependencies

1) Frameworks

   The project uses the **Ruby on Rails** web application framework. It also uses a customized (or personalized) version of **Bootstrap 4** is as a front-end framework.

2) Dependencies

   Parts of the application will make use of Wikipedia's ReST application programming interface to provide information extracts. This serves only as an addition and will not render pointless the data can be found in the database.

   The **jQuery** library is also used for sending AJAX requests for application programming interfaces and for programming some parts of the user interface.

C. The Software Stack

Software	***Version***	***Usage / Description***
Database Management System		
PostgreSQL	11	relational database management system for the data layer
Front-end		
Bootstrap	4.1.3	CSS and JS libraries for faster UI production
jQuery	3	for UI refinement and AJAX calls to backend
Backend		
Ruby	2.4.4p296	base backend programming language
(*gem*) Ruby on Rails	5.2.1	web development framework
(*gem*) pg	1.1.3	Rails plugin for interfacing with PostgreSQL
(*gem*) Puma	3.12	threaded web server
(*gem*) Sass-Rails	5.0.7	integrates Sass into Rails
(*gem*) Duktape	2.0.1.0	embeddable JavaScript engine
(*gem*) Turbolinks	5.2.0	lessens footprint of UI navigation
(*gem*) BCrypt	3.1.12	for password hashing

III. Entity-Relationship Diagram and Database Schema

A. Entities, Relationships, and the ERD

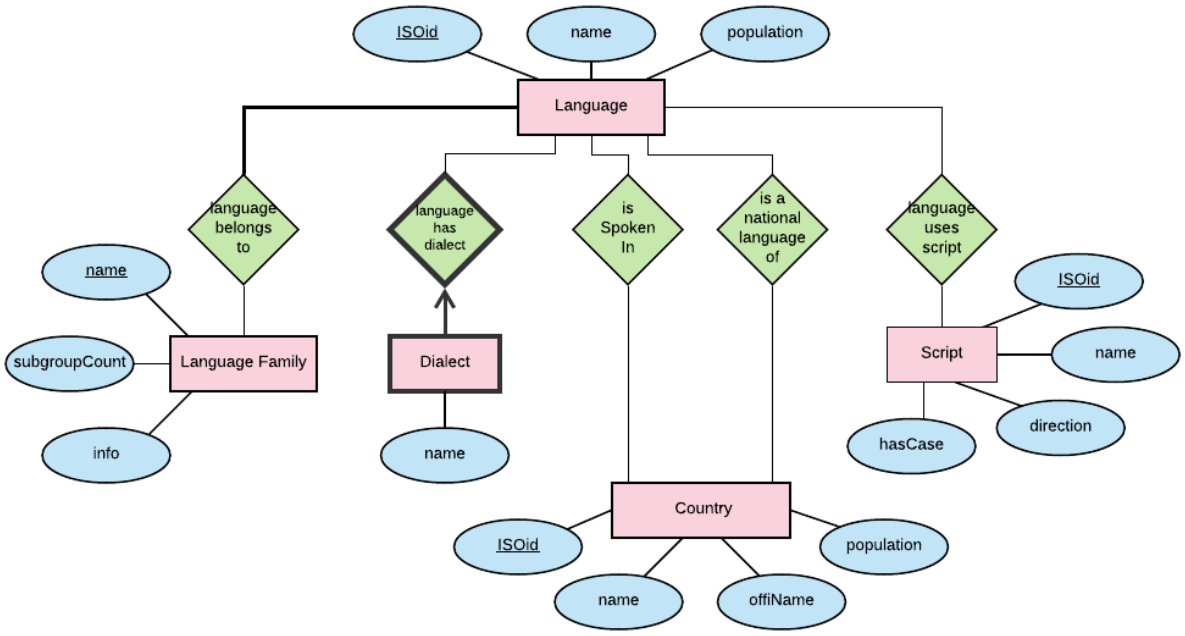
There are five (5) entity sets in the schema, and five (5) binary relationship sets that will connect the entities. These are as follows:

Entity Sets			
Entity Set Name	**Attributes**		
Language	ISOid, name, population[(1)]		
Dialect [weak entity]	*Language*.ISOid, name		
LanguageFamily	name, subgroupCount, info		
Script	ISOid, name, direction, hasCase		
Country	ISOid, name, population, offiName		
Relationship Sets			
Entity Set A	**Relationship**	**Entity Set B**	**Mapping Cardinality**
Language	languageBelongsTo	*LanguageFamily*	Many-to-One
Language	languageHasDialect	*Dialect*	One-to-Many
Language	languageUsesScript	*Script*	Many-To-Many
Language	isSpokenIn	*Country*	Many-To-Many
Language	isANationalLanguageOf	*Country*	Many-To-Many

[1] The population attribute for the *Language* entity set refers to the number of native speakers of a certain language, and has no direct correlation with the population attribute of the *Country* entity set, which refers to the population of a country as per usual context.

There is also an auxiliary entity set named *Admin* which stores username and password information. It is used for user authentication to provide a modicum of security prior to a user being given elevated access in order to manipulate the database via the user interface.

(…)



Entity-Relationship Diagram for Languages Around the World
(This diagram has been edited to better reflect the constraints in data within the database)

B. The Database Schema

Language(ISOid VARCHAR(3), name VARCHAR(50), population INTEGER)

LanguageFamily(name VARCHAR(50), subgroupCount INTEGER, info VARCHAR(3000))

(1)Dialect(*Language*.ISOid, name VARCHAR(50))

Script(ISOid VARCHAR(4), name VARCHAR(50), direction VARCHAR(15), hasCase BOOLEAN)

Country(ISOid VARCHAR(2), name VARCHAR(75), offiName VARCHAR(75) , population INTEGER)

languageBelongsTo(*Language*.ISOid VARCHAR(3), *LanguageFamily*.ISOid VARCHAR(50))

languageusesScript(*Language*.ISOid VARCHAR(3), *Script*.ISOid VARCHAR(4))

(2)isSpokenIn(*Language*.ISOid VARCHAR(3), *Country*.ISOid VARCHAR(2), isNational BOOLEAN)

[1] The *languageHasDialect* relationship identifies the weak entity *Dialect* and connects it to *Language*. Therefore, the schema representation for *languageHasDialect* would be equivalent to that of *Dialect*, and as such was not included above.

[2] The *isANationalLanguageOf* relationship in the entity-relationship diagram has been merged into *isSpokenIn* in the schema, with the Boolean isNational attribute added as an indicator for whether a language spoken in a country is of a national status or not. This approach serves to eliminate some redundancy in the database.

IV. Functionalities

A. Creates (or Inserts)

Functionality Description	Example SQL statement
Add a language to the database	`INSERT INTO Language VALUES ( 'fil', 'Filipino', 45000000 );`
Add a dialect	`INSERT INTO Dialect VALUES ( 'fil', 'Batangas' );`
Add a language family	`INSERT INTO LanguageFamily VALUES ('Austronesian', 400, 'The Austronesian language family (…)');`
Add a country	`INSERT INTO Country VALUES ( 'PH', 'The Philippines', 'Republic of the Philippines', 105000000 );`
Add a script	`INSERT INTO Script VALUES ( 'Tglg', 'Baybayin', 'left-to-right', true );`
Add a languageBelongsTo instance	`INSERT INTO languageBelongsTo VALUES ( 'fil', 'Austronesian' );`
Add a languageUsesScript instance	`INSERT INTO languageUsesScript VALUES ( 'fil', 'Tglg' );`
Add an isSpokenIn instance (national language)	`INSERT INTO isSpokenIn VALUES ( 'fil', 'PH', TRUE );`
Add an isSpokenIn instance (non-national language)	`INSERT INTO isSpokenIn VALUES ( 'ceb', 'PH', FALSE );`

B. Reads (or Selects)

Functionality Description	Example SQL statement
Get all languages	`SELECT ISOid, name FROM Language;`
Get all language families	`SELECT name FROM LanguageFamily;`
Get all scripts	`SELECT ISOid, name FROM Script;`
Get all countries	`SELECT ISOid, name FROM Country;`
Get a specific language	`SELECT * FROM Language WHERE ISOid = 'fil';`
Get a specific language family	`SELECT * FROM LanguageFamily WHERE name = 'Austronesian';`

Functionality Description	Example SQL statement

Get a specific script	`SELECT *`
	`FROM Script`
	`WHERE ISOid = ‘Tglg’;`

Get a specific country	`SELECT *`
	`FROM Country`
	`WHERE ISOid = ‘PH’;`

Get the dialects of a specific language	`SELECT name`
	`FROM Dialect`
	`WHERE ISOid = ‘fil’;`

Get the language family where a language belongs	`SELECT name`
	`FROM LanguageFamily`
	`WHERE name IN (SELECT lfamID`
	`FROM languageBelongsTo`
	`WHERE langID = 'fil');`

Get the scripts that a certain language uses	`SELECT ISOid, name`
	`FROM Script`
	`WHERE ISOid IN (SELECT scriptID`
	`FROM languageUsesScript`
	`WHERE langID = 'fil');`

Get the countries in which a certain language is a national language	`SELECT ISOid, name`
	`FROM Country`
	`WHERE ISOid IN (SELECT countryID`
	`FROM isSpokenIn`
	`WHERE langID = 'fil'`
	`AND isNational = TRUE);`

Get the countries in which a certain language is spoken	`SELECT ISOid, name`
	`FROM Country`
	`WHERE ISOid IN (SELECT countryID`
	`FROM isSpokenIn`
	`WHERE langID = 'fil');`

Get the languages that belong to a specific language family	`SELECT ISOid, name`
	`FROM Language`
	`WHERE ISOid IN (SELECT langID`
	`FROM languageBelongsTo`
	`WHERE lfamID =`
	`'Austronesian');`

Get the languages that use a specific script	`SELECT ISOid, name`
	`FROM Language`
	`WHERE ISOid IN (SELECT langID`
	`FROM languageUsesScript`
	`WHERE scriptID = 'Tglg');`

Get the national languages of a specific country	`SELECT ISOid, name`
	`FROM Languages`
	`WHERE ISOid IN (SELECT langID`
	`FROM isSpokenIn`
	`WHERE countryID = 'PH'`
	`AND isNational = TRUE);`

Get the languages spoken in a specific country	`SELECT ISOid, name`
	`FROM Languages`
	`WHERE ISOid IN (SELECT langID`
	`FROM isSpokenIn`
	`WHERE countryID = 'PH');`

Functionality Description	Example SQL statement

Search for an entry in the database (query varies per table, but always involves primary key)	`SELECT ISOid, name`
	`FROM Language`
	`WHERE name ILIKE 'PH'`
	`  OR ISOid ILIKE 'PH';`

C. Updates

Functionality Description	Example SQL statement

Edit a language's details	`UPDATE Language`
	`SET ISOid = 'eng',`
	`  name = 'English',`
	`  population = 365000000`
	`WHERE ISOid = 'fil';`

Edit a language family's details	`UPDATE LanguageFamily`
	`SET name = 'Indo-European',`
	`  subgroupCount = 107,`
	`  info = 'Indo-European languages (…)'`
	`WHERE name = 'Austronesian';`

Edit a script's details	`UPDATE Script`
	`SET ISOid = 'Latn',`
	`  name = 'Latin',`
	`  direction = 'left-to-right',`
	`  hasCase = true`
	`WHERE ISOid = 'Tglg';`

Edit a country's details	`UPDATE Country`
	`SET ISOid = 'GB',`
	`  name = 'United Kingdom',`
	`  offiName = 'United Kingdom of Great Britain and Northern Ireland',`
	`  population = 65000000`
	`WHERE ISOid = 'PH';`

Designate a language spoken in a country as national	`UPDATE isSpokenIn`
	`SET isNational = TRUE`
	`WHERE langID = 'ceb'`
	`  AND countryID = 'PH';`

Revoke the national status of a language spoken in a country	`UPDATE isSpokenIn`
	`SET isNational = TRUE`
	`WHERE langID = 'eng'`
	`  AND countryID = 'PH';`

D. Deletes

Functionality Description	Example SQL statement
Remove a language from the database	`DELETE FROM Language` `WHERE ISOid = 'eng';`
Remove a dialect	`DELETE FROM Dialect` `WHERE ISOid = 'eng' AND name = 'Scottish';`
Remove a language family	`DELETE FROM LanguageFamily` `WHERE name = 'Indo-European';`
Remove a script	`DELETE FROM Script` `WHERE ISOid = 'Latn';`

Functionality Description	Example SQL statement
Remove a country	`DELETE FROM Country` `WHERE ISOid = 'GB';`
Detach a language from its language family	`DELETE FROM belongsTo` `WHERE langID = 'eng'`

Delete a language-script relationship	`DELETE FROM languageUsesScript`
	`WHERE langID = 'eng'`
	`  AND scriptID = 'Latn';`

Detach a language as a spoken language from a country	`DELETE FROM isSpokenIn`
	`WHERE langID = 'eng'`
	`  AND countryID = 'GB';`

V. Application Access and Administrator Credentials

A. Deployment

Languages Around the World makes use of Heroku's free basic service for hosting sites. As such, the application is publicly available through https://latw.herokuapp.com. The repository for the files that make up the application can be accessed through https://github.com/dpabril/cs165dbapp.

B. Administrator Access Credentials

When an administrator is signed in, the functionalities detailed in the previous section (CRUD functionalities) which manipulate the underlying database of Languages Around the World become available through forms or dialogs in the user interface. These can be tested by the instructor should she wish to do so. The instructor's administrator credentials are as follows.

```
username     :     mjmendoza
password     :     cs165instructor!
```

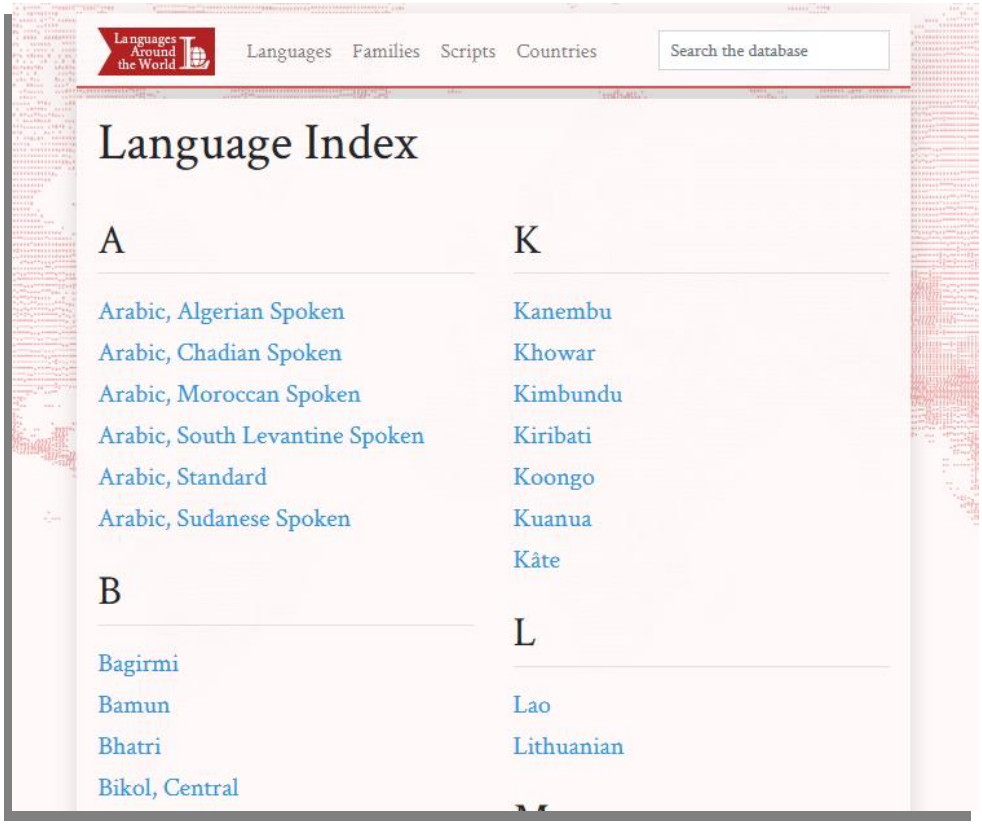
VI. Sample Screenshots

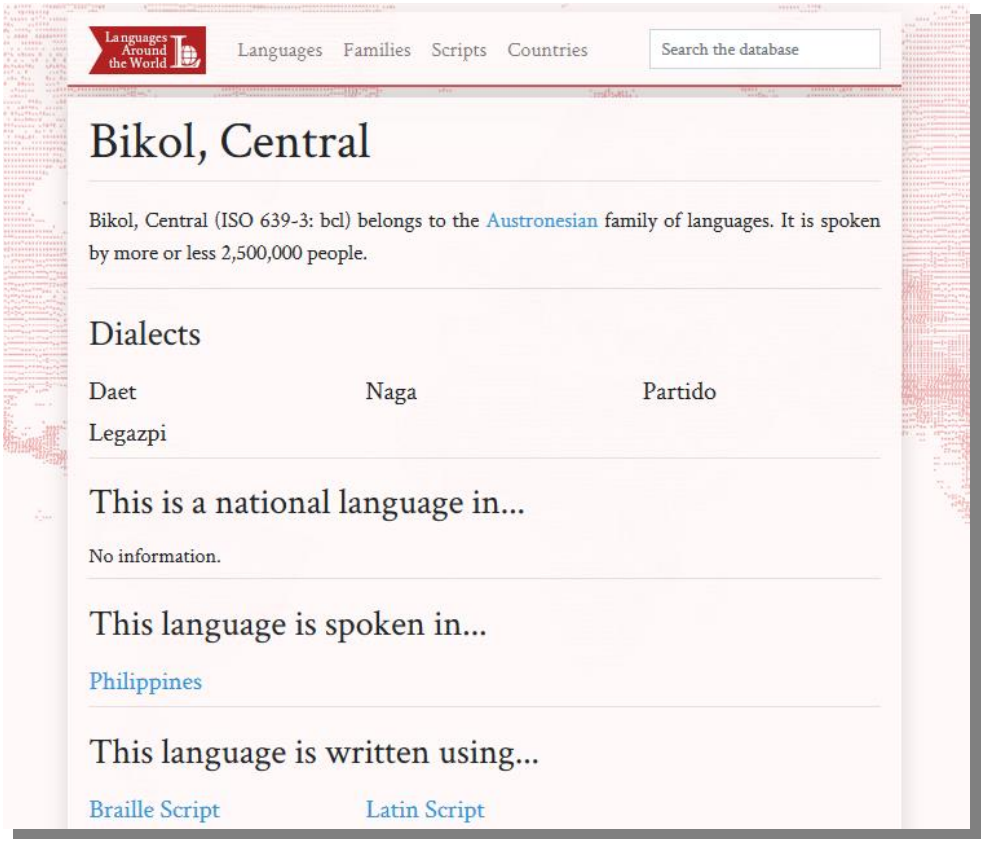
The next pages contain a few screenshots of the application. The images that can be seen here (Home and Search pages not counted) are for pages pertaining to languages in the database, but these should only be taken as templates since the pages for other entities (language families, scripts, and countries) share the same look and feel.

(…)

This is the home page of the application.

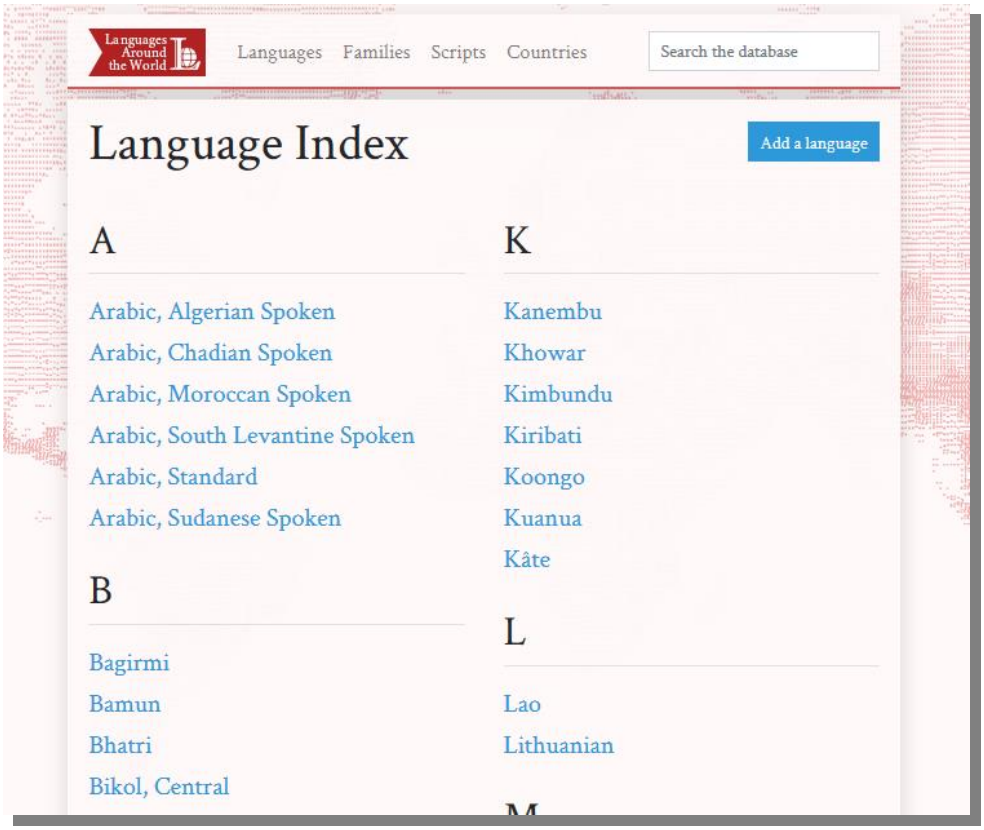
This is the page for the (language) index.





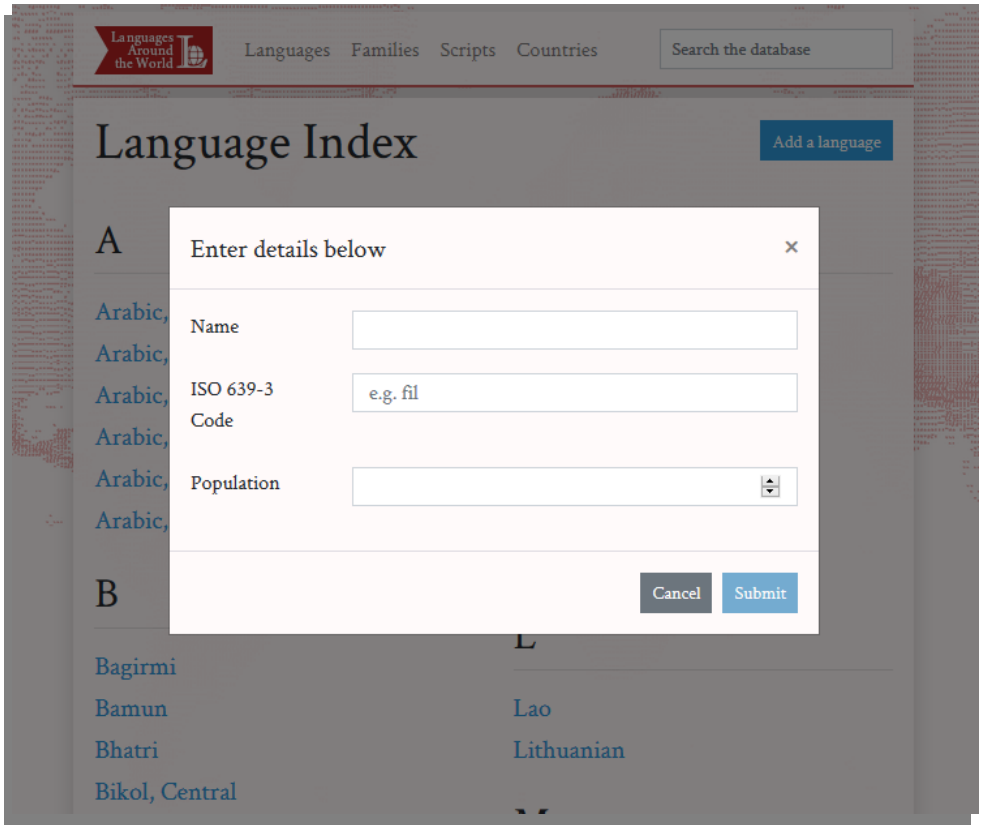
This is a page for a single (language) entry.

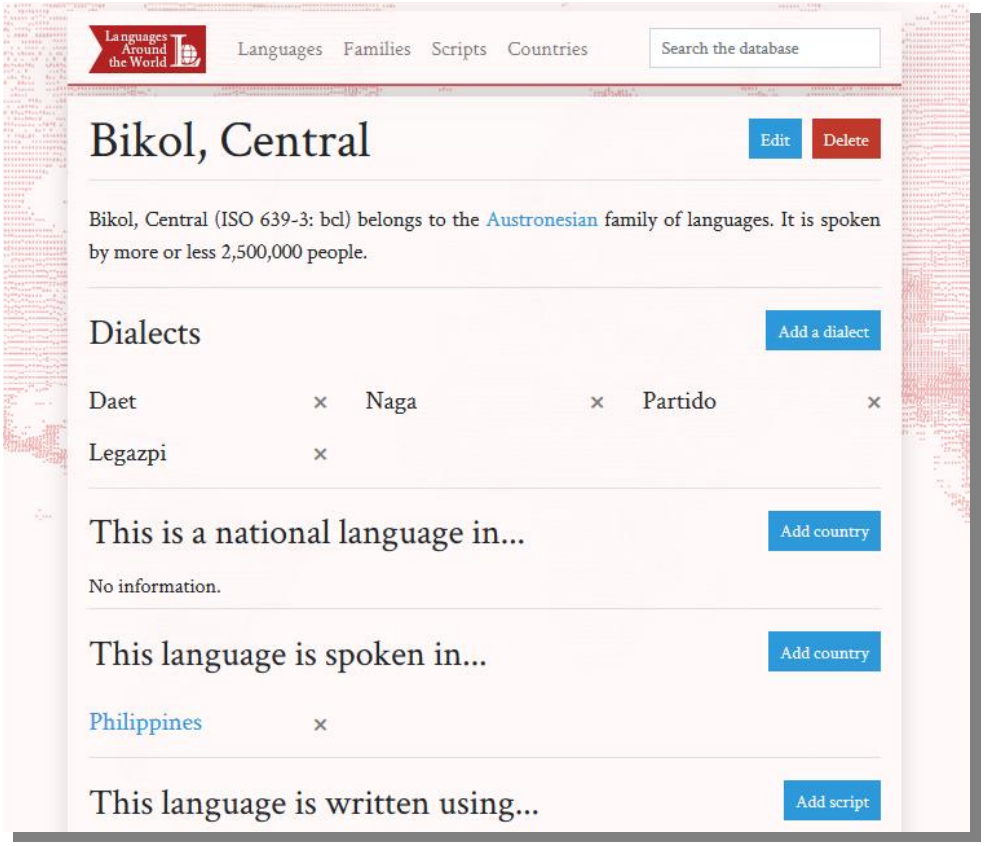
This is the search page which displays results from every table in the database (except *Admin*)



This is the **admin** page for the (language) index.

This is the creation form that pops up upon clicking the "Add a (language)" button. The form for editing entries looks similar to this.





This is the **admin** page for a single (language) entry. Note that the CRUD interface does not belong to a separate set of pages, but is integrated into the views a non-admin would see.

This is the confirmation dialog that pops up upon clicking the (language's) "Delete" button.